WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

THE USE OF THE BOXSTEP METHOD IN DISCRETE OPTIMIZATION

ROY E. MARSTEN

MARCH 1975 WP 778-75

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139

THE USE OF THE BOXSTEP METHOD IN DISCRETE OPTIMIZATION

ROY E. MARSTEN

MARCH 1975 WP 778-75

Abstract

The Boxstep method is used to maximize Lagrangean functions in the context of a branch-and-bound algorithm for the general discrete optimization problem. Results are presented for three applications: facility location, multi-item production scheduling, and single machine scheduling. The performance of the Boxstep method is contrasted with that of the subgradient optimization method.

ACKNOWLEDGEMENT

The research reported here was partially supported by National Science Foundation grants GP-36090X (University of California at Los Angeles), GJ-1154X2 and GJ-115X3 (National Bureau of Economic Research).

1. Introduction

The Boxstep method [15] has recently been introduced as a general approach to maximizing a concave, nondifferentiable function over a compact convex set. The purpose of this paper is to present some computational experience in the use of the Boxstep method in the area of discrete optimization. The motivation and context for this work is provided by Geoffrion [9] and by Fisher, Northup, and Shapiro [5,6] who have shown how the maximization of concave, piecewise linear (hence nondifferentiable) Langrangean functions can provide strong bounds for a branch-and-bound algorithm. We shall consider three applications: facility location, multi-item production scheduling, and single machine scheduling. Our experience with these applications, while limited, is quite clear in its implications about the suitability of Boxstep for this class of problems. We shall also take this opportunity to introduce two refinements of the original Boxstep method which are of general applicability.

2. The Boxstep Method

We present here a specialized version of the Boxstep method which is adequate for maximizing the Lagrangean functions which arise in discrete optimization. We address the problem

$$\max w(\pi)$$
$$\pi \geq 0 \tag{2.1}$$

where

$$w(\pi) = \min_{k \in K}(f^k + \pi g^k) \tag{2.2}$$

f^k is a scalar, π, $g^k \in R^m$, and k is a finite index set. Thus $w(\pi)$ is a concave, piecewise linear function. The Boxstep method solves (2.1) by solving a finite sequence of local problems. Using (2.2), the local problem at π^t with box size β may be written as

$$P(\pi^t;\beta) \qquad \max \sigma$$
$$\text{s.t. } f^k + \pi g^k \geq \sigma \quad \text{for } k \in K$$
$$\pi_i^t - \beta \leq \pi_i \leq \pi_i^t + \beta \text{ for } i=1, \ldots, m$$
$$\pi \geq 0$$

This local problem may be solved with a cutting plane algorithm [7,12,14]. If a global optimum lies within the current box, it will be discovered. If not, then the solution of the local problem provides a direction of ascent from π^t. The Boxstep method seeks out a global optimum as follows. (Let $\bar{P}(\pi^t;\beta)$ denote $P(\pi^t;\beta)$ with K replaced by some $\bar{K} \subseteq K$.)

Step 1. (Start) Choose $\pi^1 \geq 0$, $\varepsilon \geq 0$, $\beta > 0$. Set t=1.

Step 2. (Cutting Plane Algorithm)

(a) (Initialization) Choose $\bar{K} \subseteq K$.

(b) (Reoptimization) Solve $\bar{P}(\pi^t;\beta)$. Let $\hat{\pi}$, $\hat{\theta}$ denote an optimal solution.

(c) (Function Evaluation) Determine $k^* \in K$ such that $w(\hat{\pi}) = f^{k^*} + \hat{\pi} g^{k^*}$

(d) (Local Optimality Test) If $w(\hat{\pi}) \geq \hat{\theta} - \varepsilon$ go to step 3; otherwise set $\bar{K} = \bar{K} \cup \{k^*\}$ and return to (b).

Step 3. (Line Search) Choose π^{t+1} as any point on the ray $\{\hat{\pi} + \alpha(\hat{\pi} - \pi^t) \mid \alpha \geq 0\}$ such that $w(\pi^{t+1}) \geq w(\hat{\pi})$.

Step 4. (Global Optimality Test) If $w(\pi^{t+1}) \leq w(\pi^t) + \varepsilon$, stop. Otherwise set t=t+1 and go to Step 2.

The convergence of the method is proved in [15]. In the piecewise linear case (finite K) we may take ε=0, at least in theory. The implementation of the method works with the dual of $\bar{P}(\pi^t;\beta)$ so that new cuts can be added as new columns and the primal simplex method used for reoptimization at Step 2(b).

The motivation behind the method is the empirical observation that the number of cutting plane iterations required to solve $P(\pi^t:\beta)$ is a monotonically increasing function of β. This presents the opportunity for a trade-off between the computational work per box (directly related to β) and the number of boxes required to reach a global optimum (inversely related to β). Computational results reported in [15] demonstrate that, for a wide variety of problems, the best choice of β is "intermediate", i.e. neither very small nor very large. If β is sufficiently small, then (in the piecewise linear case) we obtain a steepest ascent method; while if β is sufficiently large, Boxstep is indistinguishable from a pure cutting

plane method. (For $\pi^1 = 0$ and $\beta = \infty$ we recover the Dantzig-Wolfe method [2].) For intermediate values of β we have something "between" these two extremes.

The three applications which follow are all of the form:

$$v^* = \min_{x \in X} f(x) \text{ s.t. } g(x) \leq b \qquad (2.3)$$

where $f : X \to k$, $g : X \to R^m$, and $X = \{x^k \mid k \in K\}$ is a finite set. The Boxstep method will be used to maximize a Lagrangean function $w(\pi)$, defined for $\pi \in R^m_+$ as

$$w(\pi) = \min_{x \in X} f(x) + \pi[g(x) - b]. \qquad (2.4)$$

Any branch-and-bound algorithm for (2.3) can compute lower bounds by evaluating this Lagrangean, since $w(\pi) \leq v^*$ for all $\pi \geq 0$. Finding the greatest lower bound, i.e. maximizing $w(\pi)$ over all $\pi \geq 0$, is a dual problem for (2.3). Thus we shall be using Boxstep to solve a Lagrangean dual of the discrete program (2.3). By defining $f^k = f(x^k)$ and $g^k = g(x^k) - b$ for all $k \in K$ we obtain the form assumed above, (2.2).

3. Facility Location with Side Constraints

The first application is a facility location model [8] of the form:

$$\min \sum_{i=1}^{m} f_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} y_{ij} \tag{3.1}$$

$$\text{s.t.} \sum_{i=1}^{m} y_{ij} = 1 \qquad j=1, \ldots, n \tag{3.2}$$

$$Ax + By \leq r \tag{3.3}$$

$$v_i x_i \leq \sum_{j=1}^{n} d_j y_{ij} \leq V_i x_i \qquad i=1, \ldots m \tag{3.4}$$

$$0 \leq y_{ij} \leq 1 \qquad \text{all } i,j \tag{3.5}$$

$$x_i = 0 \text{ or } 1 \qquad i=1, \ldots m \tag{3.6}$$

The variable x_i is an open/close variable for facility i, which will have minimum and maximum throughput v_i and V_i, respectively, if opened. The cost of opening facility i is f_i, the unit cost of serving customer j from facility i is c_{ij} and the demand of customer j is d_j. The (3.3) constraints are general linear side constraints. (In fact these will be Benders cuts since the problem displayed here is actually the master problem in a Benders decomposition context [8].) Let p denote the number of these side constraints. Geoffrion has devised a branch-and-bound algorithm for (3.1) - (3.6) which uses the Lagrangean function formed by dualizing with respect to constraints (3.2) and (3.3). If we let $(\lambda_1, \ldots, \lambda_n)$ and $(\mu_1, \ldots, \mu_p)$ be the dual variables for (3.2) and (3.3), respectively, then the Lagrangean is

$$w(\lambda,\mu) = \sum_{i=1}^{m} w^i(\lambda,\mu) - \sum_{j=1}^{n} \lambda_j + \sum_{k=1}^{p} \mu_k r_k$$

where, for each facility i,

$$w^i(\lambda,\mu) = \min_{x_i, y_{ij}} (f_i + \mu A_i)\, x_i + \sum_{j=1}^{n}{}' (c_{ij} + \lambda_j + \mu B_{ij}) y_{ij}$$

$$\text{s.t. } v_i x_i \leq \sum_{j=1}^{n} d_j y_{ij} \leq V_i x_i$$

$$0 \leq y_{ij} \leq 1 \qquad j=1, \ldots, n$$

$$x_i = 0 \text{ or } 1$$

A_i and B_{ij} are columns of A and B, respectively. Each w^i function is easily evaluated by considering the two alternatives : $x_i = 0$ and $x_i = 1$. For $x_i = 1$ we have a continuous knapsack problem.

An attempt was made to maximize $w(\lambda,\mu)$ over all $\lambda,\mu \geq 0$ with the Boxstep method. The test problem (Problem A of [8]) has m=9 facilities, n=40 customers, and p=7 side constraints (Benders cuts). Thus $\pi = (\lambda,\mu) \in R^{47}$. Problem (3.1)-(3.5) with (3.6) replaced by $(0 \leq x_i \leq 1)$ was solved as a linear program and the optimal dual variables for constraints (3.2) and (3.3) are taken as the starting values for λ and μ, respectively. At Step 2(a), $\bar{K}$ is taken as all cuts already generated, if any. The line search at Step 3 is omitted ($\pi^{t+1} = \hat{\pi}$); the tolerance is $\varepsilon=10^{-6}$. Table 1 reports the outcome of four runs, each of 20 seconds' duration (IBM360/91). The last four columns give the number of $w(\pi)$ evaluations, number of linear programming pivots, the pivot/evaluation ratio, and the value of the best solution found.

The results are not encouraging. Convergence of the first local problem could not be achieved for a box size of .25, .10, or .01. Convergence was finally achieved with $\beta = .001$ and 8 local problems were completed in the 20 seconds. The increase in the Lagrangean over these 8 boxes amounted to 76% of the difference between the starting value (10.595676) and the global optimum (10.850098). The price paid for this increase, in terms of

computation time, is prohibitively high, however. Geoffrion [8] has executed an entire branch-and-bound algorithm for this problem in under 2 seconds (same IBM360/91). Geoffrion did not attempt to maximize the Lagrangean in his algorithm but simply used it to compute strong penalties [9].

Table 1. Facility Location Problem

β	boxes	w(π) eval	LP piv	piv/eval	best
.250	<1	37	745	20.1	10.595676*
.100	<1	34	706	20.7	10.595676*
.010	<1	43	433	10.1	10.793467
.001	8	58	282	4.9	10.789490

*starting value
'global optimum at 10.850098

The computational burden on the cutting plane algorithm for a given local problem $P(\pi^t;\beta)$ depends on the number of cuts needed and on the average number of LP pivots required to reoptimize after a cut is added. This average is given by the pivot/evaluation ratio and is recorded in Table 1. In the present application, difficulty was encountered with both of these factors. First, some cuts had no effect on the objective function value, θ. As many as ten successive cuts had to be added before the value of θ dropped. This is simply a reflection of degeneracy in the dual of $P(\pi^t;\beta)$. The effect of this degeneracy is to increase the number of cuts needed for convergence. The second and more serious difficulty, however, is the great number of pivots (more than 20 for $\beta \geq .10$) required for each reoptimization. This is in marked contrast to other applications where, typically, only one or two pivots are required. See section 4 below and the results in [15]. This behavior was quite unexpected and appears to be a kind of instability. Starting with only one negative reduced cost coefficient (for the newly introduced cut), each pivot eliminates one negative but also creates one (or more). This process continues for several pivots before optimality is finally regained. Unfortunately this phenomenon is not unique to this class of facility location problems but arises in the application of section 5 as well. Its effect is to impose a heavy "overhead" on the Boxstep method, rendering it very expensive computationally.

Three suggestions that might be offered are: (a) generate a separate column for each facility at each iteration [12, p. 221] (b) use the dual simplex method at Step 2 (b); and (c) use a larger tolerance, say $\varepsilon=10^{-3}$. The outcomes are : (a) much worse; (b) much worse; and (c) no change. We shall return to this test problem in section 6.

4. Multi-item Production Scheduling

The second application we shall consider is the well-known Dzielinski-Gomory multi-item production scheduling model with one shared resource [3,12,13]. Two test problems are used: one with I=25 items and T=6 time periods, the other with I=50 and T=6. The variables $\pi = (\pi_1, \ldots, \pi_T)$ are the prices of the shared resource in each time period; resource availability in each period is given by $b = (b_1, \ldots, b_T)$. The Lagrangean function $w(\pi)$ is given by

$$w(\pi) = \sum_{i=1}^{I} w^i(\pi) - \sum_{k=1}^{T} \pi_k b_k \qquad (4.1)$$

where $w^i(\pi)$ is the optimal value of a single-item production scheduling problem of the Wagner-Whitin type [16] and is evaluated by a dynamic programming algorithm. Thus evaluating $w(\pi)$ involves solving I separate T-period dynamic programs.

The 25- and 50-item problems are solved, for several box sizes, using Boxstep. The origin is taken as the starting point at Step 1 (π^1=0) and the line search is omitted at Step 3 ($\pi^{t+1} = \hat{\pi}$). No more than 13 cuts are carried. (Once 13 cuts have been accumulated, old non-basic cuts are discarded at random to make room for new ones.) A tolerance of $\varepsilon = 10^{-6}$ is used. Note that $\pi \varepsilon R^6$. The results are presented in Tables 2 and 3. For each run the number of $w(\pi)$ evaluations, LP pivots, and pivot/evaluation ratio are recorded. The computation times are in seconds for an IBM370/165. For the 25-item problem w(0) =47,754.00 and $v^* = w(\pi^*) = 48,208.80$; while for the 50-item problem w(0) = 92,602.00 and $v^* = w(\pi^*) = 94,384.06$.

In this application the Boxstep method has no difficulty in reaching a global optimum. Notice that the pivot/evaluation ratio never exceeds 2. This is a significant qualitative difference from the facility location problem of section 2. Examination of local problem convergence reveals the signs of degeneracy in the dual of $P(\pi^t;\beta)$, that is, several cuts may be required to reduce δ. This difficulty can apparently be overcome (at least in R^6) as long as each reoptimization takes only one or two pivots.

The same two test problems were also solved by the direct Generalized Upper Bounding (GUB) approach advocated by Lasdon and Terjung [13]. The times are 2.20 seconds and 6.87 seconds for the 25-item and 50-item problems, respectively. This suggests that Boxstep may be quite successful on Dzielinski-Gomory problems, particularly since these usually involve only a T = 6 or 12 month horizon. This will require testing on industrial-size problems for verification (e.g. I=400, T=12).

These production scheduling problems will serve to illustrate a refinement of the original Boxstep method. Let $\hat{\pi}$ denote an optimal solution of the local problem $P(\pi^t;\beta)$. We may define $G^t = w(\hat{\pi}) - w(\pi^t)$ as the gain achieved in box t. Because of the concavity of $w(\pi)$, we would expect the gain achieved in successive boxes to decline as we approach a global optimum. For example, in the β = .20 run from Table 2, the sequence of gains over the nine boxes is: 271, 71, 33, 25, 23, 14, 10, 6, 2 (rounded). Notice that solving the first local problem gives us some idea of the gain to be expected in the second. Since solving a local problem to completion is often not worth the computational cost when we are far from a global optimum, this suggests the following cutoff rule. Choose an "anticipated gain" factor γ, $0 < \gamma \leq 1.0$, and if while working on $P(\pi^{t+1};\beta)$ a point $\tilde{\pi}$ is generated with

$$w(\tilde{\pi}) \geq w(\pi^{t+1}) + \gamma G^t,$$

then stop the cutting plane algorithm, set $\hat{\pi} = \tilde{\pi}$, and proceed immediately to Step 3. (In this event take $G^{t+1} = G^t$.) A large value of γ should have little effect on the trajectory $\{\pi^t \mid t=1,2, \ldots\}$ while offering the possibility of computational savings. Too small a value of γ, however, may cause wandering in response to small improvements and hence an increase in the number of boxes required. These effects may be observed in Table 4 where the $\beta = .10$ and $\beta = .20$ runs from Table 2 are repeated with alternative gain factors (γ=1 reproduces the original results). The column headed "subopt" gives the number of local problems which are terminated when the anticipated gain is achieved. In both cases the maximum reduction in computation time is a little less than 40%.

Table 2. Twenty-five item problem; original Boxstep method.

β	boxes	v(y) eval's	LP pivots	piv/eval	time
.10	18	98	105	1.1	2.81
.20	9	85	104	1.2	2.45
.30	6	70	99	1.4	2.23
.40	5	68	93	1.4	2.02
.50	4	77	111	1.4	2.20
.60	3	56	72	1.3	1.57
.70	3	65	81	1.2	1.85
.80	3	61	86	1.4	1.80
.90	2	44	64	1.5	1.21
1.00	2	53	74	1.4	1.42
1.25	2	49	67	1.4	1.27
1.50	2	54	78	1.4	1.45
1.75	1	32	47	1.5	0.86
2.00	1	36	56	1.6	0.96
10.00	1	42	58	1.4	1.10
20.00	1	50	75	1.5	1.31

Table 3. Fifty item problem; original Boxstep method.

β	boxes	v(y) eval's	LP pivots	piv/eval	time
1.0	8	187	229	1.2	7.05
2.0	4	130	154	1.2	4.82
5.0	2	99	126	1.3	3.59
10.0	1	68	103	1.5	2.71
20.0	1	71	96	1.4	2.77
40.0	1	72	97	1.3	2.83

Table 4. Twenty-five item problem; suboptimization based on anticipated gain factors (γ).

β	γ	boxes	subopt	v(y) eval	LP pivots	time
10	.10	19	16	64	93	1.72
10	.30	18	11	66	91	1.77
10	.50	18	11	77	105	2.05
10	.80	18	2	99	114	2.52
10	1.00	18	0	98	105	2.81
20	.10	11	8	60	89	1.57
20	.30	9	5	62	80	1.55
20	.50	9	4	69	84	1.78
20	.80	9	1	77	91	2.06
20	1.00	9	0	85	104	2.45

5. Single Machine Scheduling.

Finally we consider the single machine scheduling model of Fisher [4]*. The problem is to schedule the processing of n jobs on a single machine so as to minimize total tardiness. Job i has processing time p_i, due date d_i, and start time x_i (all integer valued). To obtain bounds for a branch-and-bound algorithm, Fisher constructs the Lagrangean function

$$w(\pi) = \min_{x \in X} \sum_{j=1}^{n} \{\max \{x_j + p_j - d_j, 0\} + \sum_{k=x_j+1}^{x_j+p_j} \pi_k\}$$

where π_k is the price charged for using the machine in period k and X is a finite set determined by precedence constraints on the starting times. Fisher, who has devised an ingenious special algorithm for evaluating $w(\pi)$, uses the subgradient optimization method [11] to maximize $w(\pi)$.

When using subgradient optimization the sequence of Lagrangean values $\{w(\pi^t) \mid t = 1, 2, \ldots\}$ is not monotonic and there is no clear indication of whether or not a global optimum has been found. Consequently, a predetermined number of steps is made and the biggest $w(\pi)$ value found is taken as an approximation of the maximum value of the Lagrangean. It was therefore of interest to determine how close the subgradient optimization method was coming to the true maximum value. To answer this question, one of these Lagrangeans was maximized by the Boxstep method.

A second refinement of the original Boxstep method is illustrated in this application. An upper limit is placed on the number of cutting plane

---

*The author is grateful to Marshall Fisher for his collaboration in the experiments reported in this section.

iterations at Step 2. If this limit is exceeded, then the local problem $P(\pi^t;\beta)$ is terminated and the box is contracted (set $\beta = \beta/E$ for $E > 1$). Furthermore, if $\tilde{\pi}$ is the best solution of $P(\pi^t;\beta)$ generated so far, and $w(\tilde{\pi}) > w(\pi^t)$, then we take $\pi^{t+1} = \tilde{\pi}$; otherwise $\pi^{t+1} = \pi^t$. This provides another opportunity for suboptimizing local problems and also offers some automatic adjustment if the initial box size is too large.

The test problem used is taken from [4] and has $n = 20$ jobs. The number of time periods is the sum of all n processing times, in this case 53. Thus $\pi \in R^{53}$. The starting point for Boxstep is the best solution found by the subgradient optimization method. Furthermore, some of the subgradients that are generated are used to supply Boxstep with an initial set of linear supports. (If $w(\tilde{\pi}) = f^* + \tilde{\pi}g^*$, then g^* is a subgradient of $w(\pi)$ at $\pi = \tilde{\pi}$.)

For the 20-job test problem, subgradient optimization took about one second (IBM360/67) to increase the Lagrangean from $w(0) = 54$ to $w(\pi^1) = 91.967804$. The Boxstep method was started at π^1 with $\beta = 0.1$. Up to 55 cuts were carried and a tolerance of $\epsilon = 10^{-6}$ was used. A maximum of 10 cutting plane iterations was allowed for each local problem. Each contraction replaced the current β by $\beta/2$. These parameters ($\beta = 0.1$, 55 cuts, 10 iterations, $E = 2$) were chosen after some exploratory runs had been made.

The final run is summarized in Table 5. Four boxes were required to reach the global optimum, $v^* = w(\pi^*) = 92$. The first two of these boxes had to be contracted; the last two converged. The time for Boxstep was 180 seconds. As with the facility location problem, this is exceedingly

expensive. Fisher [4] reports that the entire branch-and-bound algorithm for this problem took only 1.8 seconds. The details of this run display the same two phenomena we have encountered before: a high pivot/evaluation ratio (as in section 3) and degeneracy in the dual of $P(\pi^t\beta)$ (as in sections 3 and 4).

Table 5. Single Machine Scheduling Problem

	β	$w(\pi)$ eval	LP piv	piv/eval
Box 1	.0100	10	168	16.8
Box 2	.0050	10	82	8.2
Box 3	.0025	7	75	10.7
Box 4	.0025	1	20	20.0

6. Conclusions

The most promising alternative method for maximizing the class of Lagrangean functions we have considered here is subgradient optimization [10,11]. Subgradient optimization tends to produce a close approximation to the global maximum, v*, for a very modest computational cost. Fortunately, this is precisely what is needed for a branch-and-bound algorithm. Since v* is not actually required, the time spent pursuing it must be weighed against the enumeration time that can be saved by having a tighter bound. This is dramatically illustrated in the example of section 5. Subgradient optimization obtained $w(\pi^1) = 91.967804$ in about one second. Since it is known that the optimal value of the problem is integer, any $w(\pi)$ value can be rounded up to the nearest integer, in this case 92. Boxstep spent 180 seconds verifying that 92 was indeed the global maximum. This is a factor of 10^2 longer than the 1.8 seconds required for the complete branch-and-bound algorithm!

To further illustrate this qualitative difference, the performance of Boxstep and subgradient optimization was compared on the facility location problem of section 3. An approximate line search was used at Step 3 of the Boxstep method and suboptimization of the local problems was done as in section 4, with $\gamma = 1/2$. The box size was held fixed at $\beta = .001$ and up to 56 cuts were carried. The global maximum was found at $w(\pi^*) = 10.850098$ after a sequence of 28 local problems and line searches. This required 318 $w(\pi)$ evaluations, 929 LP pivots, and over 90 seconds of CPU time (IBM370/168). The subgradient optimization method, starting from the same initial solution, reached the global maximum (exactly) in only 0.9 seconds-again a factor of 10^2! This required only 75 steps ($w(\pi)$ evaluations). It is apparent

from these and other results [4,5,11] that subgradient optimization is the preferred method in this context. Boxstep may be viewed as a method "last resort" to be used if it is essential to find an exact global maximum. In this event, Boxstep can start from the best solution found by subgradient optimization and can be primed with an initial set ($\bar{K} \subseteq K$) of subgradients.

The performance of the Boxstep method is clearly limited by the rate of convergence of the imbedded cutting plane algorithm. Wolfe [17] has provided an invaluable insight into the fundamental difficulty we are encountering. He shows that for a strongly and boundedly concave function (as our Lagrangeans would typically be), the convergence ratio is at best of $(a/4A)^{1/2n}$ where $0 < a \leq A$ and n is the dimension of the space. Notice that the convergence ratio gets worse (i.e. approaches unity) quite rapidly as n increases. The Boxstep method attempts to overcome this slow convergence by imposing the box constraints, thereby limiting the number of relevant cuts (indices $k \in K$). What we observe when n is large, however, is that to achieve even near convergence the box must be made so small that we are forced into an approximate steepest ascent method. (Boxstep can do no worse than steepest ascent, given the same line search, since it is based on actual gain rather than initial rate of gain.) Steepest ascent is already known to work very poorly on these problems [5].

Degeneracy in the dual of the local problem $P(\pi^t;\beta)$ is an important characteristic of all of the problems we have considered. This is not surprising since this dual is a convexification of the original problem (2.3) and degeneracy in the linear programming approximations of discrete problems is a well-known phenomenon. The effect of this degeneracy is to further slow the convergence of the cutting plane algorithm. In two of the three applications we have encountered the phenomenon of

high pivot/evaluation ratios. That is, many LP pivots are required to reoptimize after each new cut is added. This difficulty, when present, increases the computational burden associated with each cut. It is not clear yet whether this is caused by problem structure or is another consequence of higher dimensionality.

There remains one opportunity which we have not investigated here. In the course of a branch-and-bound algorithm we have to solve many problems of the form (2.1). The Lagrangean function is somewhat different in each case, but the optimal π-vector may be nearly the same. When this is the case, starting Boxstep at the previous optimal π-vector and using a small box can produce rapid detection of the new global optimum. This has recently been applied with considerable success by Austin and Hogan [1].

References

[1] L.M. Austin and W.W. Hogan, "Optimizing Procurement of Aviation Fuels", Working Paper, U.S. Air Force Academy (June 1973). (To appear in *Management Science.*)

[2] G.B. Dantzig and P. Wolfe, "Decomposition Principles for Linear Programs", *Operations Research* 8 (1) (1960) 101-111.

[3] B.P. Dzielinski and R.E. Gomory, "Optimal Programming of Lot Sizes, Inventory, and Labor Allocations", *Management Science* 11 (1965) 874-890.

[4] M.L. Fisher, "A Dual Algorithm for the One-Machine Scheduling Problem", Technical Report No. 243, Department of Operations Research, Cornell University (December 1974).

[5] M.L. Fisher, W. Northup, and J.F. Shapiro, "Using Duality to Solve Discrete Optimization Problems: Theory and Computational Experience", *Mathematical Programming*, Study 3.

[6] M.L. Fisher and J.F. Shapiro, "Constructive Duality in Integer Programming", *SIAM J. Appl. Math.*, 27 (1) (1974).

[7] A.M. Geoffrion, "Elements of Large-Scale Mathematical Programming", *Management Science* 16 (11) (July 1970) 652-691.

[8] A.M. Geoffrion, "The Capacitated Facility Location Problem with Additional Constraints", Graduate School of Management, University of California at Los Angeles, (February 1973).

[9] A.M. Geoffrion, "Lagrangean Relaxation for Integer Programming", *Mathematical Programming*, Study 2, December, 1974, 82-114.

[10] M. Held and R.M. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees: Part II", *Mathematical Programming* (1) (1971) 6-25.

[11] M. Held, P. Wolfe, and H. Crowder, "Validation of Subgradient Optimization", *Mathematical Programming*, 6 (1) (1974) 62-88.

[12] L.S. Lasdon, *Optimization Theory for Large Systems* (The Macmillan Company, New York, 1970).

[13] L.S. Lasdon and R.C. Terjung, "An Efficient Algorithm for Multi-Item Scheduling", *Operations Research*, 19 (4) (1971) 946-969.

[14] D.G. Luenberger, *Introduction to Linear and Nonlinear Programming* (Addison-Wesley, Reading, Mass., 1973).

[15] R.E. Marsten, W.W. Hogan, and J.W. Blankenship, "The Boxstep Method for Large Scale Optimization", *Operations Research*, 23 (3) (1975).

[16] H.M. Wagner and T.M. Whitin, "A Dynamic Version of the Economic Lot Size Model", *Management Science* 5 (1958) 89-96.

[17] P. Wolfe, "Convergence Theory in Nonlinear Programming", in: *Integer and Nonlinear Programming*, Ed. J. Abadie (North Holland, Amsterdam, 1970), 1-36.

Appendix 1. Data for the facility location problem.
(m=9, p=7, n=40)

i	f_i	v_i	V_i
1	.069837	.600000	1.709064
2	.065390	.600000	1.823961
3	.072986	.600000	1.316324
4	.068788	.600000	2.163716
5	.072986	.600000	2.203619
6	.064241	.600000	1.959994
7	.067739	.600000	2.691376
8	0.0	.153312	3.063312
9	0.0	.271873	2.638127

i	d_i	i	d_i
1	.081426	21	.088804
2	.206631	22	.084573
3	.027246	23	.159997
4	.303292	24	.292932
5	.044033	25	.100370
6	.034300	26	.104647
7	.128681	27	.079968
8	.093232	28	.032030
9	.024378	29	.185120
10	.254841	30	.136249
11	.337371	31	.386822
12	.173633	32	.031088
13	.229527	33	.028759
14	.066465	34	.526940
15	.114898	35	.245977
16	.054371	36	.121745
17	.114093	37	.060440
18	.178713	38	.038508
19	.087465	39	.113301
20	.136781	40	.072122

Let $C=(C^1, C^2, \ldots, C^9)$ where $C^i = (c^i_j)$ for $i=1, \ldots, 9$; and $j=1, \ldots, 40$. Only the finite components of C will be listed. (Each facility can serve only a subset of 40 customers).

c^1_j	j	c^2_j	j	c^3_j	j
.146312	1	.177966	1	.415698	13
.335647	2	.400414	2	.203779	15
.057449	3	.066692	3	.196776	17
.576832	4	.651596	4	.167838	21
.076995	5	.088610	5	.280627	23
.069982	6	.079259	6	.446496	24
.208937	7	.261337	7	.193918	25
.199051	8	.206829	8	.207758	26
.048757	9	.052407	9	.157175	27
.485979	10	.443745	10	.063178	32
.699711	11	.698083	11		
.358736	12	.344807	12		
		.232256	15		

c_i^4	i	c_i^5	i	c_i^6	i
.211549	26	.730936	24	.373499	2
.180904	27	.223619	25	.067868	3
.077770	28	.216805	26	.559347	4
.418627	29	.179398	27	.091838	5
.312908	30	.817279	31	.075631	6
.690737	31	.058582	32	.208560	8
.061964	32	.068050	33	.047310	9
.065836	33	.898408	34	.371875	13
1.051362	34	.529652	35	.119268	14
.562520	35	.277590	36	.188485	15
.306933	36	.134402	37	.094147	16
.138593	37	.103633	38	.221074	17
.092578	38	.303734	39	.344947	18
.274659	39	.233171	40	.162663	19
.221286	40.			.267876	20
				.154611	22
				.317310	23

c_i^7	i	c_i^8	i	c_i^9	i
.169821	1	.417318	13	.155846	1
.358741	2	.125278	14	.352344	2
.062991	3	.105156	16	.062468	3
.523075	4	.229567	17	.261840	20
.088090	5	.323004	18		
.070213	6	.164339	19		
.288279	7	.162281	21		
.205946	8	.149220	22		
.047654	9	.308354	23		
.546177	10	.607881	24		
.807097	11	.236317	25		
.407905	12	.214657	26		
.407497	13	.164406	27		
.125318	14	.066657	28		
.201701	15	.337940	29		
.097901	16	.354037	30		
.241546	17				
.354386	18				
.158701	19				
.247122	20				

Let $B = [B^1, B^2, \ldots, B^9]$ where $B^i = (b^i_{pj})$ for $i = 1, \ldots, 9$; $p=1, \ldots, 7$; and $j=1, \ldots, 40$. Only the non-zero components of B will be listed. Note that $B^8 = 0$ and $B^9 = 0$.

B^1: $b^1_{2j} = -1.0$ for $j = 5,7,8,10,11,12$

B^2: $b^2_{3j} = -1.0$ for $j = 10,11,12$

B^3: $b^3_{5j} = -1.0$ for $j = 17,23,24,25$

B^4: $b^4_{6j} = -1.0$ for $j = 31,38,39$

B^5: $b^5_{7j} = -1.0$ for $j = 34,35,36,37$

B^6: $b^6_{4j} = -1.0$ for $j = 13,14,16$

B^7: $b^7_{1j} = -1.0$ for $j = 4,16,19$

A

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
1						1.0	4.0		
2	7.0	1.0							
3	1.0	4.0							
4						4.0	1.0		
5			4.0						
6				3.0					
7					4.0				

r = (1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0)

Appendix 2. Data for the multi-item production scheduling problem.

We shall use the following notation [12, pp. 171-177].

s_i = set-up cost for item i,

h_i = inventory holding cost for item i,

p_i = unit production cost for item i,

a_i = amount of resource required for one set-up for item i,

k_i = amount of resource required to produce one unit of item i,

b_t = amount of resource available in time period t.

D_{it} = demand for item i in period t.

The data for the 50-item problem will be given. The first 25 items constitute the 25-item problem when used with the resource vector b^{25} = (3550,3550,3250, 3250, 3100, 3100). The resource vector for the full 50-item problem is b^{50} = (7000,6000,6000,6000,6000,6000). Both problems have 6 time periods. Let $h_i = 1$, $p_i = 2$, and $k_i = 1$ for all i=1, ..., 50. Let [x] denote the largest integer that does not exceed the real number x, and let $\emptyset(j) = j \pmod 5$ for any integer j. Then for i=1, ..., 25 we have

$$a_i = 10 * \left\{ \left[\frac{i-1}{5} \right] + 1 \right\}$$

$$s_i = 75 + 25 * \emptyset(i-1)$$

while for i=26, ..., 50 we have

$$a_i = 5 + 10 * \emptyset(i-26)$$

$$s_i = 30 * \left\{ \left[\frac{i-26}{5} \right] + 2 \right\}.$$

DEMAND

i	D_{i1}	D_{i2}	D_{i3}	D_{i4}	D_{i5}	D_{i6}
1	120	113	97	88	96	103
2	100	125	78	116	97	84
3	147	99	108	77	162	141
4	58	95	136	154	91	113
5	112	118	64	132	154	104
6	62	121	133	98	96	63
7	103	106	124	114	126	95
8	96	96	98	103	94	100
9	112	106	141	106	107	111
10	86	85	91	103	114	96
11	100	109	98	94	85	86
12	125	130	120	90	100	120
13	110	140	103	89	85	110
14	103	102	106	121	93	96
15	90	94	99	108	117	124
16	87	93	92	109	105	108
17	112	84	76	92	105	100
18	100	120	130	102	97	88
19	65	73	102	114	91	106
20	150	100	75	50	125	100
21	92	120	79	92	107	103
22	99	126	133	85	122	106
23	100	114	136	102	105	105
24	101	103	124	93	91	90
25	77	86	99	104	121	134
26	125	138	84	102	103	52
27	86	81	92	120	110	94
28	104	106	63	41	120	113
29	100	90	126	87	77	96
30	131	152	186	142	130	120
31	114	100	130	85	63	128
32	96	56	88	100	115	115
33	102	52	77	63	75	115
34	88	100	94	74	140	120
35	74	53	82	91	51	115
36	16	43	72	86	103	110
37	95	104	130	100	103	85
38	107	120	96	100	116	120
39	124	124	97	99	120	125
40	62	64	87	41	124	133
41	72	98	73	114	135	166
42	23	41	55	58	99	130
43	123	163	180	100	84	93
44	107	104	117	104	87	93
45	32	102	74	42	130	152
46	121	117	138	104	142	100
47	150	103	84	100	110	112
48	56	64	79	81	89	90
49	69	84	107	48	63	132
50	100	107	120	101	95	88

Appendix 3. Data for the single machine scheduling problem.

job i	P_i	d_i
1	6	34
2	10	61
3	5	56
4	1	23
5	9	80
6	9	1
7	1	18
8	5	21
9	2	14
10	3	113
11	7	95
12	4	77
13	6	63
14	2	56
15	3	60
16	8	78
17	10	1
18	6	58
19	9	27
20	8	24